https://wiki.teltonika-networks.com/view/Teltonika_EYE_device_pairing_and_data_sender_configuration_example

Teltonika EYE device pairing and data sender configuration example

The information in this page is updated in accordance with firmware version **00.07.03.2**.

☐

Contents

Introduction and prerequisites

In this article, we will explore the usage of Teltonika Eye sensors and beacons with Teltonika Networks devices. The Teltonika Eye sensor is a powerful tool that allows for monitoring and tracking of vehicles and other mobile assets. When paired with Teltonika devices, the sensor can provide a wealth of information such as identification, temperature, humidity and even movement.

For this use case, we are going to need Teltonika Networks device with **Bluetooth functionality**, for example, RUTX11. We are also going to need either a **Teltonika EYE sensor** or **Teltonika EYE beacon**.

Connection topology

Connection topology in this case is simple, we are going to send Bluetooth data from either **EYE sensor** or **EYE beacon** using Data to server functionality. More detailed topologies are presented in the configuration sections.

Adjusting EYE device protocol and settings

The Teltonika Eye sensor supports several Bluetooth protocols, including iBeacon and Eddystone. By supporting both Protocols, the Teltonika Eye sensor can be used in a variety of different applications and environments. For more information regarding these settings, please refer to this article: EYE App configuration

Connecting Bluetooth sensors to the device

Follow these steps to pair either EYE sensor or EYE beacon to the RUTX11 or another Teltonika device of your choice.

Navigate to **Services -> Bluetooth** and follow the steps below:

1. Press on the toggle to **Enable Bluetooth**
2. Press **Scan**

Once the Scan is finished, you should be able to see a list with all Bluetooth devices around. You can identify your EYE device by looking at the MAC address on top of it. To Pair an EYE device to our RUTX11, simply select it **(1)** and press Pair **(2)**.

If the EYE device has paired successfully, you should be able to find it in the **Paired Devices** list. Press on the **Details** button to view if the details and protocol are displayed correctly.

Here is how the information should be displayed if you connect Bluetooth EYE Sensor and press the **Details** button. You should be able to see information like Humidity, Temperature, Signal Strength, Battery Voltage, etc.

Bluetooth Data to Server via MQTT protocol

In this section, we are going to upload Bluetooth data to server using MQTT Protocol. Here is the connection topology to help you understand things better:

Prerequisites for this example:

- MQTT Broker
- MQTT Subscriber

You could set up MQTT Broker on Teltonika device, or use an external service as a broker. In order to subscribe to the topic, we are going to use Mosquitto application.

Data sender setup

Follow these steps to configure Data to server functionality:

- Navigate to **Services -> Data to server**
- Press **Add** to add new data sender

In the next screen, adjust settings accordingly:

1. **Enable**: on
2. **Name**: BT_MQTT (input any preferred name)
3. **Data source**: Bluetooth data
4. **Protocol**: MQTT
5. **JSON format**: {"Data": "%b", "Hour": "%d"} - adjust JSON format according to your needs
6. **Segment count**: 1
7. **URL/Host/Connection string**: 192.168.1.139 (MQTT broker address)
8. **Topic**: BT_data

Leave everything else as default and press **Save & Apply**

MQTT Subscriber setup and testing

Follow these steps to subscribe on the **BT_data** topic and get the Bluetooth data via MQTT:

- Navigate to your **Mosquitto** installation folder
- While holding **Shift** key, use **Mouse right-click** an press **Open Powershell window here**

Once open, input this command to subscribe on the topic:

```
.\mosquitto_sub -t BT_data -h 192.168.1.139
```

Here **-t** option means topic and **-h** is used to set MQTT broker IP address.

If everything is set up correctly, you should be able to see incoming MQTT messages:

Bluetooth Data to Server using HTTP protocol

In this section, we are going to upload Bluetooth data to server using HTTP Protocol. Here is the connection topology to help you understand things better:

We are going to use a Hercules app as a TCP server to get Bluetooth sensor data from the device.

Data sender setup

Follow these steps to configure Data to server functionality:

- Navigate to **Services -> Data to server**
- Press **Add** to add new data sender

In the next screen, adjust settings accordingly:

1. **Enable**: on
2. **Name**: BT_HTTP (input any preferred name)
3. **Data source**: Bluetooth data
4. **Protocol**: HTTP
5. **JSON format**: {"Data": "%b", "Hour": "%d"} - adjust JSON format according to your needs
6. **Segment count**: 1
7. **URL/Host/Connection string**: 192.168.1.211:8080 (server address and port)

Leave everything else as default and press **Save & Apply**

TCP server setup and testing

Since we are using Hercules application as TCP Server, the setup is simple: start the application and switch to the TCP Server tab. Make sure to listen on the same port **(1)** as set up in data sender configuration (**8080** for this example) and you should see the Bluetooth data incoming **(2)**.

Summary

This configuration example demonstrates the process of pairing Teltonika Bluetooth Eye Sensors and Beacons with a RUTX11 router. The setup also includes testing the functionality of Bluetooth data transmission to a server using both HTTP and MQTT protocols. The example provides a step-by-step guide on how to connect the sensors and beacons to a router and effectively send data to a remote server for analysis and processing.

References

Teltonika EYE sensor - More information about Teltonika EYE sensor

Teltonika EYE beacon - More information about Teltonika EYE beacon

RUTX11 Data to server - More information about Data to Server functionality

EYE App configuration - Teltonika EYE App configuration

Mosquitto - Mosquitto MQTT Client

Hercules - Hercules TCP Client / Server application